ECSE 420 Assignment 1

Connor Fowlie 260687955 & Antoine Khouri 260683888

Question 1.6

Graph 1.4: Runtime decreases until a certain point. Afterwards, it continually increases. This makes sense, as after a certain point, the overhead generated from creating and switching threads causes diminishing returns, and eventually a decrease in performance (past the optimal thread count). This can be explained by the fact that there can only be so many actual physical threads, and past a certain point the threads being created and used are virtual, which creates more and more overhead.

Graph 1.5: The sequential multiplication run time is of O(n^3), therefore it increases greatly as n (here: matrix size) increases. However for n rather small (<=500) the overhead generated by creating and switching threads during parallel multiplication is large enough to cause sequential multiplication to run faster. However as n increases, the parallelization of the process allows the parallel multiplication's runtime to increase at a much slower rate than O(n^3), causing it to eventually be slower than sequential multiplication�s runtime.

Question 2.2

Deadlock can be avoided in multiple ways with this example:

```
- 1: Forcing threads to release resources after certain time to allow other request
to complete
- 2: Requiring an order of resources that ensures no deadlock (must have A to
request B)
- 3: Deadlock detection and avoidance (See Bankers algorithm)
```

Question 4

4.1

The limit would be reached if we theoretically have infinite processors working on the parallel section. This would result in the parrallel section (60%) happening instantly, leaving only 40% of the processing time as the maximum possible speed up. This results in a speed up maximum of 2.5x.

4.2

Sn = 1/((1-p)+p/n). Furthermore S'n>2*Sn => S'n>2/((1-p)+p/n)

S'n=1/((1-p/k)+p/(n*k))

<=> 1/((1-p/k)+p/(n*k))> 2/((1-p)+p/n) Flip => 1-p/k+p/(n*k)<1/2(1-p+p/n)

p=0.8

<=> 2-2_0.8/k+2_0.8/(n*k)<1-0.8+0.8/n

Multiply by k:

<=> 2k-1.6+1.6/n<k-0.8k+0.8k/n*

<=> 2k-0.2k+0.8k/n<1.6-1.6/n

<=> k(1.8+0.8/n)<1.6-1.6/n

<=> k<(1.6-1.6/n)/(1.8+0.8/n)

4.3

Let a1 be the original acceleration and a2 the second. Since the program is half as fast, we know that a2/a1=2

A1=1/((1-p)+p/n)=q/(s+(1-s)/n) A2=1/((s/3)+(1-s/3)/n)

a2/a1=2

<=> 2=(1/(s/3+(1-s/3)/n)/(1/(s+(1-s)/n))

<=> 2*(1/(s+(1-s)/n))=(1/(s/3+(1-s/3)/n)

<=> 2s/3+2/n-2s/3n=s+(1-s)/n

<=> 2sn +6 -2s = 3sn +3-3s

<=> 2sn-3sn-2s+3s=-3

<=> -sn+s=-3

<=> s(n-1)=3

<=> s=3/(n-1)